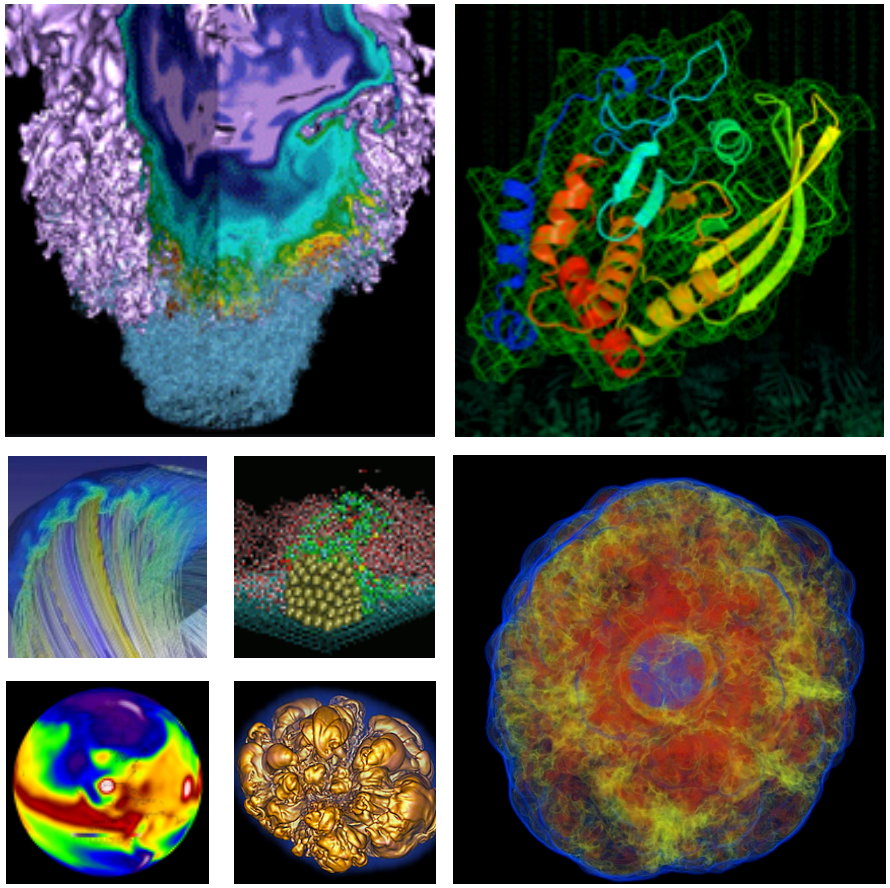
BLAST Training



Daniel Udwary

NERSC Data Science Engagement Group

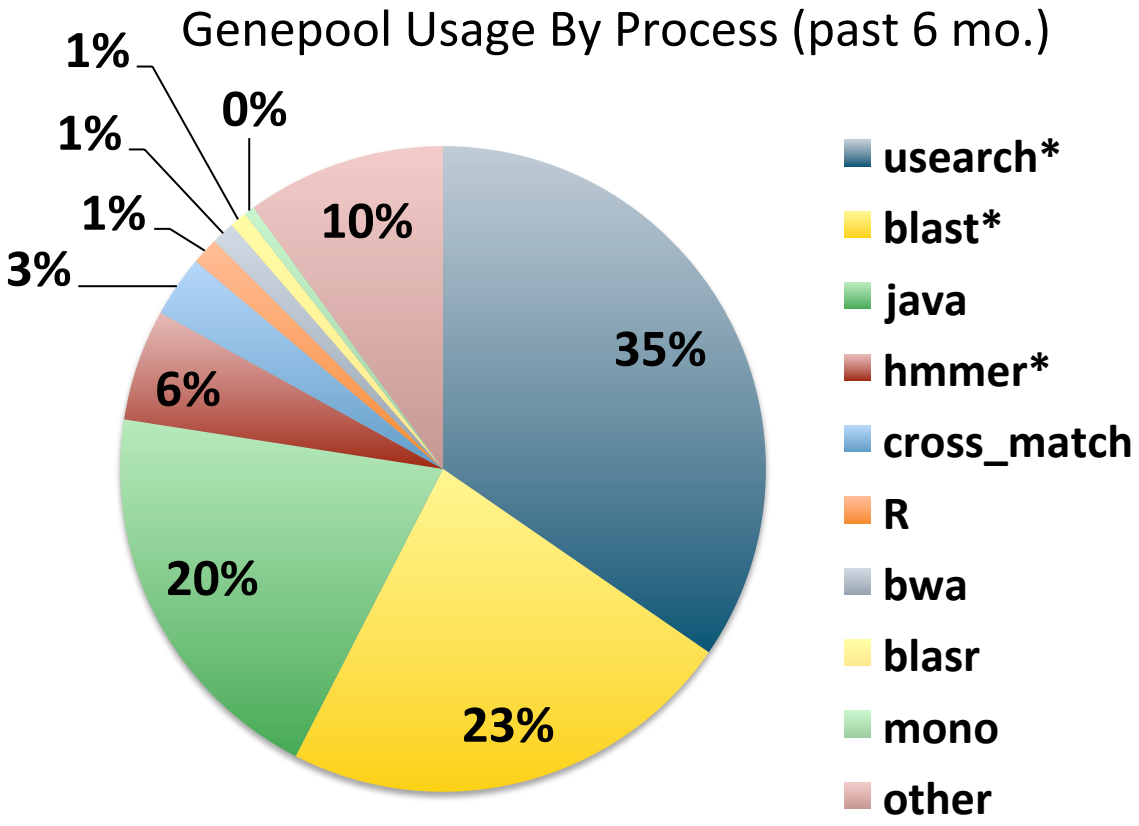
December 16, 2015

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

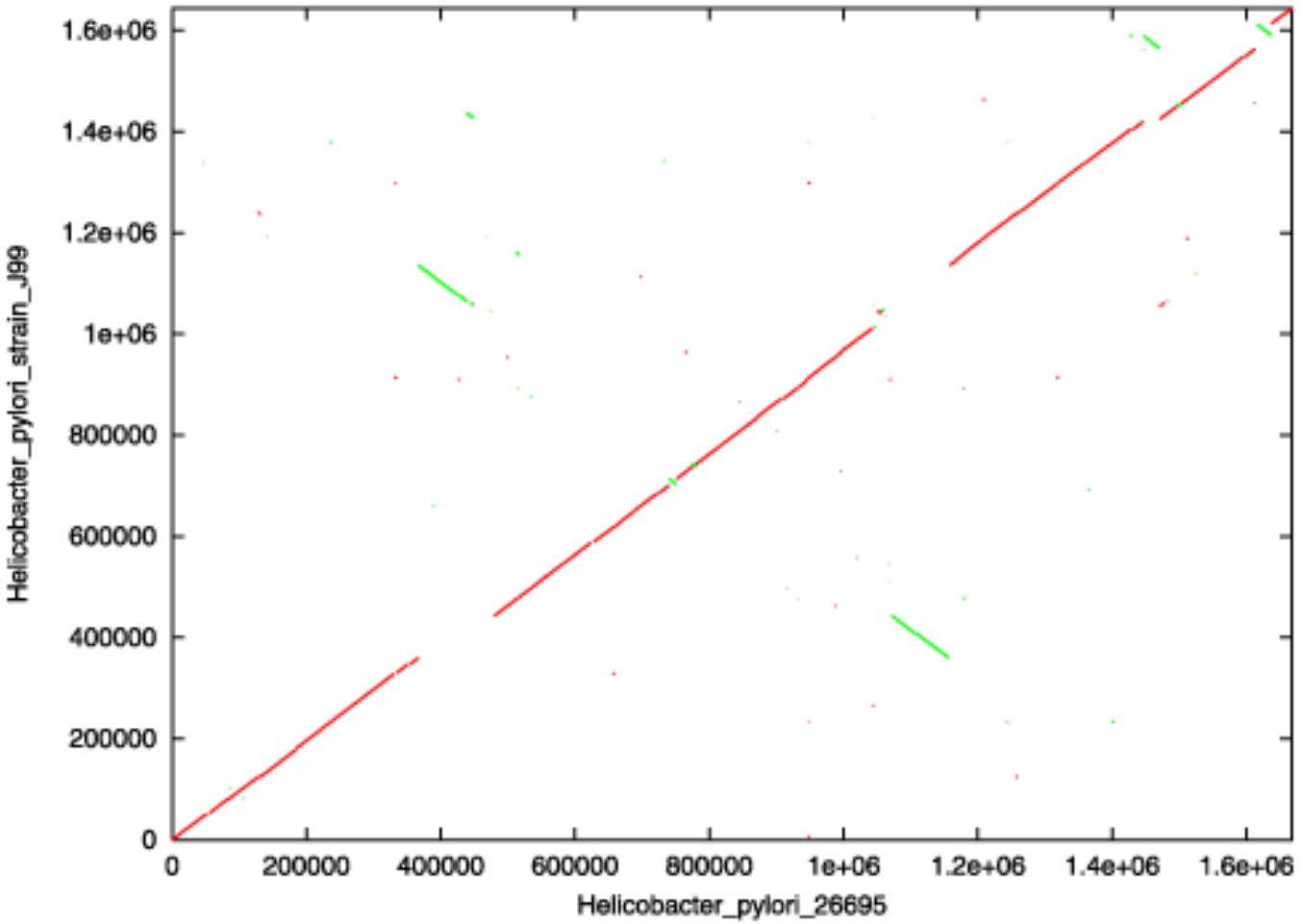
Why am I running a BLAST tutorial?

BLAST makes up a very significant portion of the JGI's compute workload



And, we've found that most BLAST usage on Genepool could be done more efficiently…

Simple dotplots



How it works

Compare my name: Daniel Wayne Udwary
With my father's: Wayne John Udwary

	D	A	N	I	E	L	W	A	Y	N	E	U	D	W	A	R	Y
W																	
A																	
Y																	
N																	
E																	
J																	
O																	
H																	
N																	
U																	
D																	
W																	
A																	
R																	
Y																	

How it works

Compare my name: Daniel Wayne Udwary
With my wife's: Megan Eileen Welsh

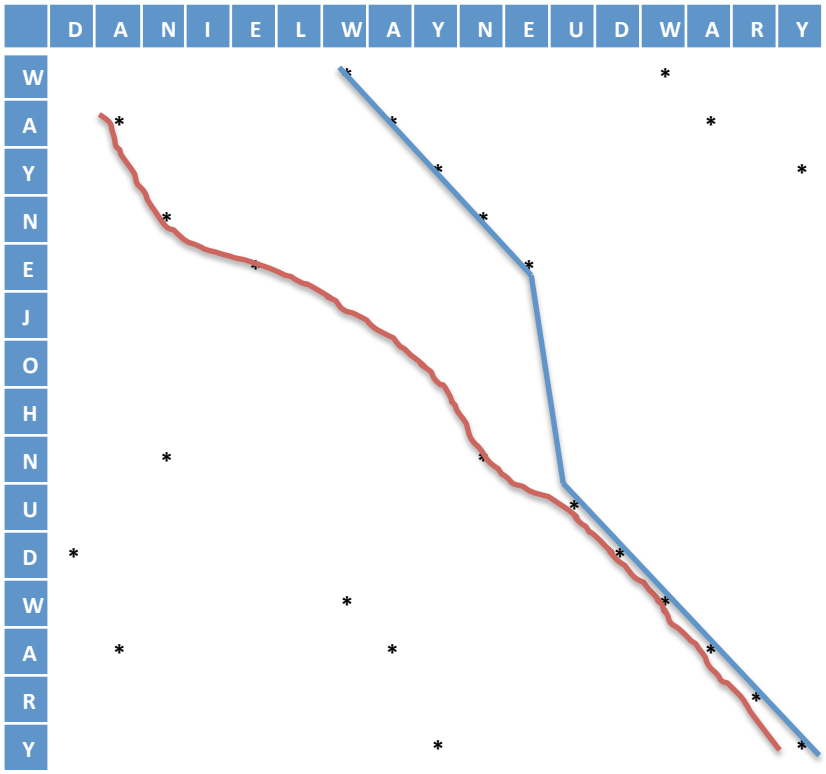
	D	A	N	I	E	L	W	A	Y	N	E	U	D	W	A	R	Y
M																	
E					*						*						
G																	
A		*						*							*		
N			*							*							
E					*						*						
I				*													
L						*											
E					*						*						
E					*						*						
N			*							*							
W														*			
E					*						*						
L						*											
S																	
H																	

Local vs Global Alignment

Which is “best”?

```
DANIELWAYNE----UDWARY
......|||||....||||||
------WAYNEJOHNUDWARY
```

```
DA-NIELWAYNEUDWARY
.|.|.|....|.||||||
WAYN-E-JOHN-UDWARY
```



BLAST searches for the best local alignment(s)

“Basic Local Alignment Search Tool”

Recommendation Zero:

- **Is BLAST the best tool for the job?**
- **Which BLAST executable is going to give you the results you actually want?**

blastn
blastp
blastx
tblastn
tblastx
deltablast
legacy_blast.pl
psiblast
rpsblast
rpstblastn

Computation time:
tblastx >> blastx > blastp > blastn

Possible alternatives:
LAST
BLAT
MegaBLAST
hmmer
usearch
short-read aligners
phylogenetic tools

The BLAST algorithm

- **Four major steps:**
  - Seeding
    - Drawing the dots
  - Extension
    - Drawing the lines
  - Evaluation
    - Figuring out which lines are best
  - Output
    - Writing the results out to disk

Seeding

- Significant alignments have “words” in common.

When comparing two sequences, BLAST breaks sequences into each word

- Each word and its location stored in a database prior to execution of the search.
- So, word search/matching is just database lookup. (Very fast)

Problems (depending on word size)

-- But some words too common to be “important”

-- Some related sequences may have no common words

So, BLAST employs a “neighborhood” of similar words to generate “word hits” if score *T* threshold is reached.

Word	Score (T)
RGD	17
KGD	14
QGD	13
RGE	13
EGD	12
HGD	12
NGD	12
RGN	12
AGD	11
MGD	11
RAD	11
RGQ	11
RGS	11
RND	11

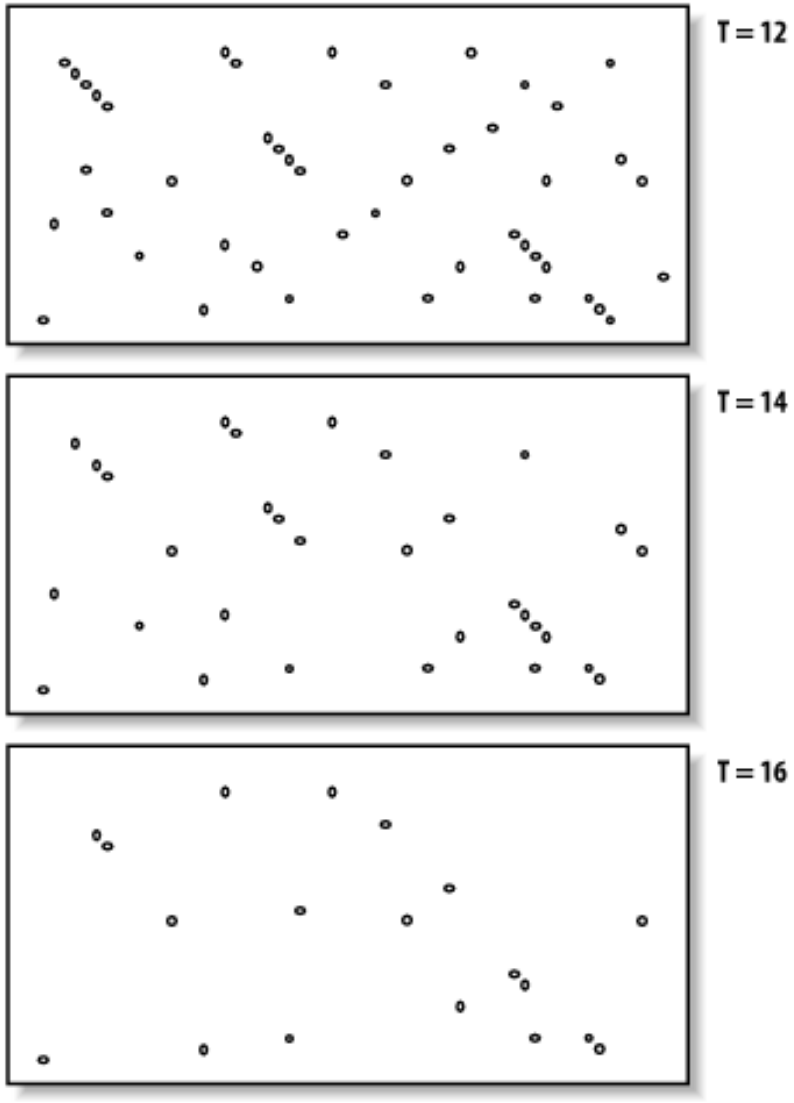
Next, generate locations of “word hits” between the two sequences

Can adjust word size and *T* threshold to balance sensitivity with speed.

Higher *T* = fewer word hits = faster algorithm

Protein sequence alignments:
-- Generally word size = 2 or 3
-- For speed, often *T* is set arbitrarily large
-- Filtering of “low-complexity” regions
(ie repetitive sequences)

Some differences when aligning nucleotides:
-- Use only exact-match words (ie no *T*)
-- Adjust sensitivity with word size



Extension

Word hits are extended to find additional matching regions

```
The quick brown fox jumps over the lazy dog.
The quiet brown cat purrs when she sees him.
```

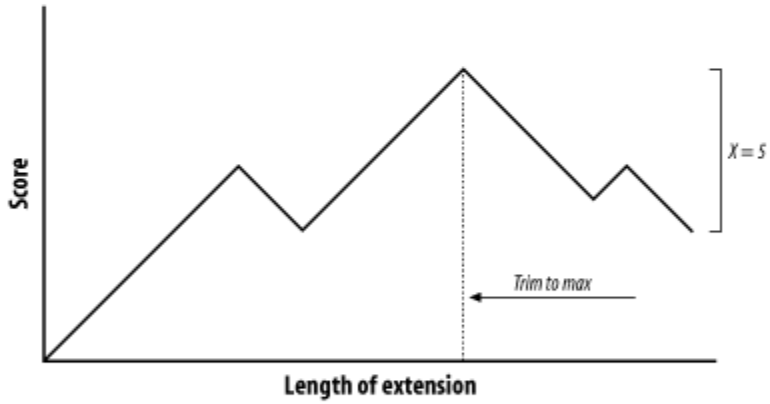
Assume 'T' in 'The' is the seed, and we extend to the right.

```
The quic
The quie
```

First mismatch. Do we continue or not?

Create variable *X*, that represents how much the score is allowed to drop off since the last maximum. So, for *X*=5:

```
The quick brown fox jump
The quiet brown cat purr
123 45654 56789 876 5654 <- score
000 00012 10000 123 4345 <- drop off score
```



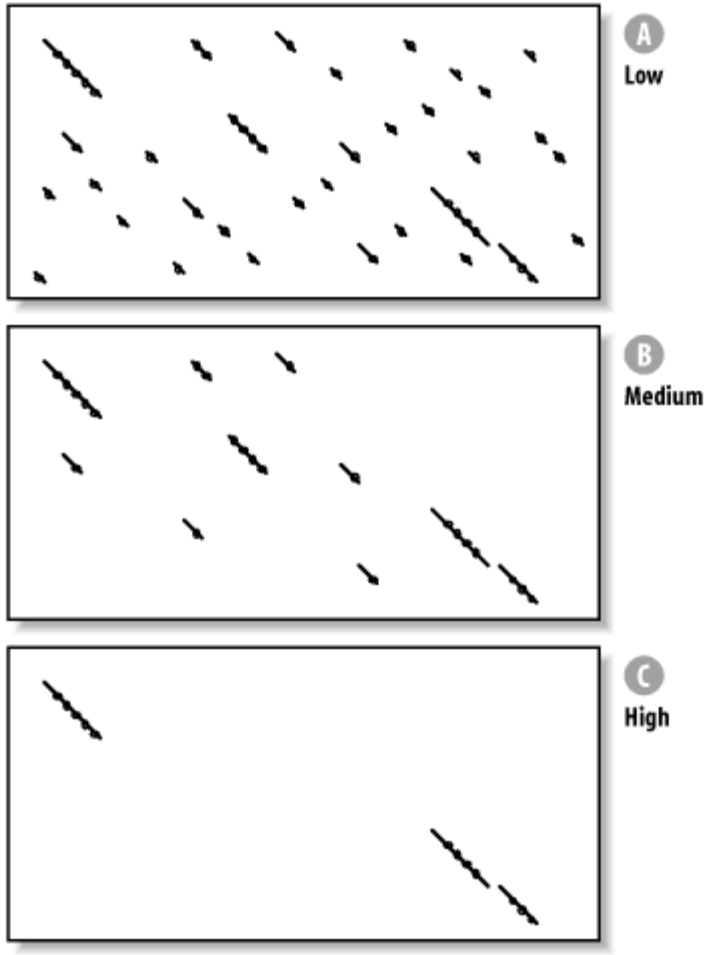
Note: Originally, gaps not considered.
Modern implementations use scoring systems with gap penalties considered.

Evaluation

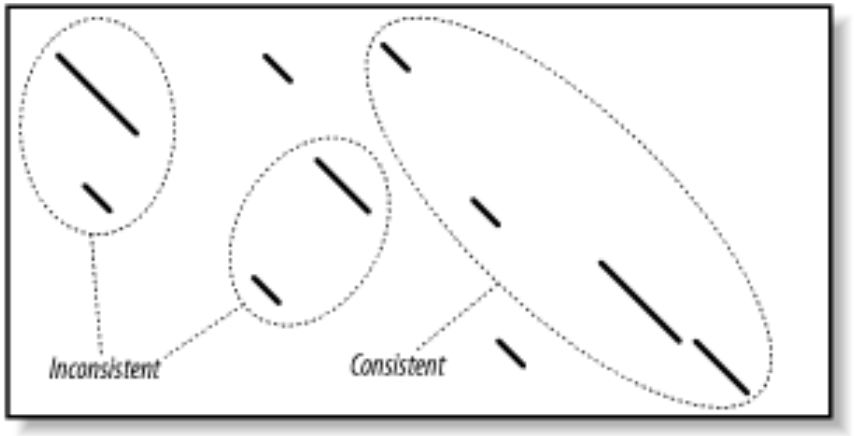
Determine if alignment(s) are statistically significant
- HSP = High-scoring Segment Pair

Goal: Set an alignment threshold score to eliminate small HSPs that may not be relevant.

Common thresholds:
- Certain number of top hits (default = 500)
- % identity (default = not used)
- Evalue (default = 10)

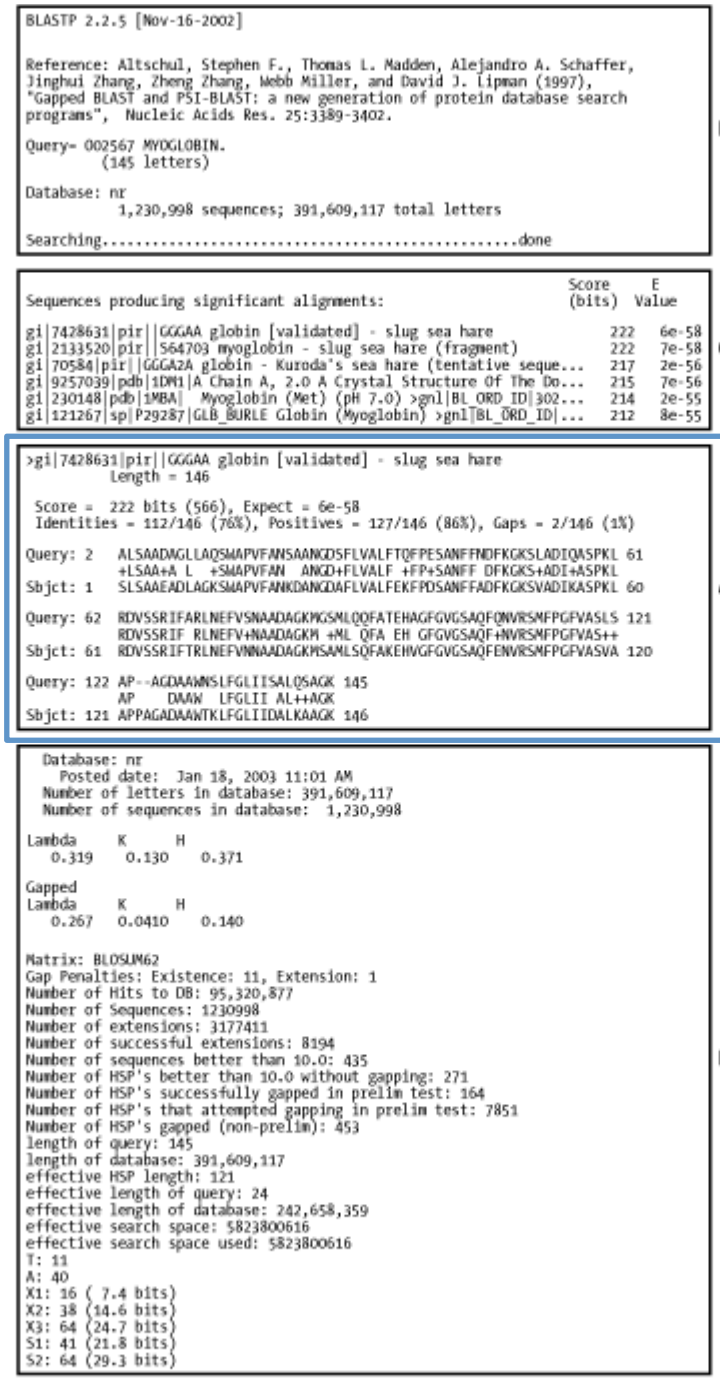


Alignments must also be “consistent”



ie always “downhill”

BLAST output



```
BLASTP 2.2.5 [Nov-16-2002]

Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= 002567 MYOGLOBIN.
         (145 letters)

Database: nr
           1,230,998 sequences; 391,609,117 total letters

Searching.....................................................done
```

Header

```
                                                                 Score     E
Sequences producing significant alignments:                      (bits)  Value

gi|7428631|pir||GGGAA globin [validated] - slug sea hare             222   6e-58
gi|2133520|pir||S64703 myoglobin - slug sea hare (fragment)          222   7e-58
gi|70584|pir||GGGA2A globin - Kuroda's sea hare (tentative seque...  217   2e-56
gi|9257039|pdb|1DM1|A Chain A, 2.0 A Crystal Structure Of The Do...  215   7e-56
gi|230148|pdb|1MBA|  Myoglobin (Met) (pH 7.0) >gnl|BL_ORD_ID|302...  214   2e-55
gi|121267|sp|P29287|GLB_BURLE Globin (Myoglobin) >gnl|BL_ORD_ID|...  212   8e-55
```

One-line summaries

```
>gi|7428631|pir||GGGAA globin [validated] - slug sea hare
          Length = 146

 Score =  222 bits (566), Expect = 6e-58
 Identities = 112/146 (76%), Positives = 127/146 (86%), Gaps = 2/146 (1%)

Query: 2   ALSAADAGLLAQSWAPVFANSAANGDSFLVALFTQFPESANFFNDFKGKSLADIQASPKL 61
           +LSAA+A L  +SWAPVFAN  ANGD+FLVALF +FP+SANFF DFKGKS+ADI+ASPKL
Sbjct: 1   SLSAAEADLAGKSWAPVFANKDANGDAFLVALFEKFPDSANFFADFKGKSVADIKASPKL 60

Query: 62  RDVSSRIFARLNEFVSNAADAGKMGSMLQQFATEHAGFGVGSAQFQNVRSMFPGFVASLS 121
           RDVSSRIF RLNEFV+NAADAGKM +ML QFA EH GFGVGSAQF+NVRSMFPGFVAS++
Sbjct: 61  RDVSSRIFTRLNEFVNNAADAGKMSAMLSQFAKEHVGFGVGSAQFENVRSMFPGFVASVA 120

Query: 122 AP--AGDAAWNSLFGLIISALQSAGK 145
           AP    DAAW  LFGLII AL++AGK
Sbjct: 121 APPAGADAAWTKLFGLIIDALKAAGK 146
```

Alignments

```
  Database: nr
    Posted date:  Jan 18, 2003 11:01 AM
  Number of letters in database: 391,609,117
  Number of sequences in database:  1,230,998

Lambda     K      H
   0.319    0.130    0.371

Gapped
Lambda     K      H
   0.267   0.0410    0.140

Matrix: BLOSUM62
Gap Penalties: Existence: 11, Extension: 1
Number of Hits to DB: 95,320,877
Number of Sequences: 1230998
Number of extensions: 3177411
Number of successful extensions: 8194
Number of sequences better than 10.0: 435
Number of HSP's better than 10.0 without gapping: 271
Number of HSP's successfully gapped in prelim test: 164
Number of HSP's that attempted gapping in prelim test: 7851
Number of HSP's gapped (non-prelim): 453
length of query: 145
length of database: 391,609,117
effective HSP length: 121
effective length of query: 24
effective length of database: 242,658,359
effective search space: 5823800616
effective search space used: 5823800616
T: 11
A: 40
X1: 16 ( 7.4 bits)
X2: 38 (14.6 bits)
X3: 64 (24.7 bits)
S1: 41 (21.8 bits)
S2: 64 (29.3 bits)
```

Footer

Standard format
- human readable, but not very easy to parse

-outfmt <String>
  alignment view options:
    0 = pairwise,
    1 = query-anchored showing identities,
    2 = query-anchored no identities,
    3 = flat query-anchored, show identities,
    4 = flat query-anchored, no identities,
    5 = XML Blast output,
    6 = tabular,
    7 = tabular with comment lines,
    8 = Text ASN.1,
    9 = Binary ASN.1,
   10 = Comma-separated values,
   11 = BLAST archive format (ASN.1),
   12 = JSON Seqalign output,
   13 = JSON Blast output,
   14 = XML2 Blast output

The *modern* BLAST algorithm

- **Four major steps:**
  - Seeding
    - Break sequence(s) into words, search for similar words in database
    - Calculate E-value to select HSPs
    - Pass selected number of HSPs to extension step
  - Extension
    - Generate alignment
  - Evaluation
    - Perform additional evaluations (%identity, %positives, %gaps)
  - Output
    - Write out to disk in one of 16 formats

Recommendation 1:

- **Use a modern BLAST**
  - "module load blast+" = most recent NCBI BLAST package
    - v2.2.31+
  - "module load blast" = "legacy" blast, deprecated in 2007
    - v2.2.26
  - Different executables, slightly different outputs
    - So, we keep the old one around for older software/pipelines
    - Don't use legacy blast, if you can at all help it!!!
  - BLAST+ also shows much better parallelization
    - (much more on that in a few slides!)

Recommendations 2 & 3

- **Use a realistic E-value threshold**
  - "-evalue
  - Know what it is you want to know from your results.
  - In large databases (nt, nr) anything above 1E-5 is firmly in the "Twilight Zone" of sequence homology.
    - Default of 10 is just asking for lots of extra compute time and meaningless results
  - For annotation, 1E-20 probably better for interpreting functional relationships

- **Set a sequence output limit**
  - "-max_target_seqs 10" for top ten hits. Top 5?
  - Does anyone ever want 500 results?

(There may need to be a balance between these two settings…)

A quick thought on Genepool and parallelization…

I need to run 1000 blast searches. So, I'll just set up a task array of 1000 jobs, and it will go a lot faster!

Then, imagine this picture with ~1000 bowls, and 140,000 puppies per day

Putting parallelization to the test

Annotation simulation

- Task: use blastp to compare 100 “putative” protein sequences against NCBI refseq_protein database
- First 100 protein sequences from *Streptomyces coelicolor*

2.2.26 “Legacy” blastall –p blastp
vs
2.2.31 “Blast+” blastp

Contaminant detection simulation

- Task: use blastn to compare 100 1kb “contig” sequences from a bacterial plasmid against NCBI nt database
- 1kb sequences from bacterial plasmid

2.2.26 “Legacy” blastall –p blastn
vs
2.2.31 “Blast+” blastn

Putting parallelization to the test

"Annotation" test

Executable	input	threads/ job	Evalue th	output sequences	time to results (not counting queue time)	min/max walltime per job	total walltime	walltime hours
2.2.26	1x100	1	10	500/250	28 m	95/1599	81034	22.5
2.2.26	1x100	1	1.00E-20	500/250	37 m	218/2195	71806	19.9
2.2.26	1x100	1	1.00E-20	5/5	34 m	195/2053	59674	16.6
2.2.31+	1x100	1	1.00E-20	5	26 m	208/1558	59734	16.6
2.2.31+	100x1	1	0.1	500	595 m		35731	9.9
2.2.31+	100x1	8	0.1	500	90 m		5364	1.5
2.2.31+	100x1	16	0.1	500	84 m		5061	1.4
2.2.31+	100x1	16	1.00E-20	500	36 m		2164	0.6

Putting parallelization to the test

"Contaminant detection" test

100 sequences blasted vs NCBI "nt" database with default parameters, except specification of number of threads

Executable	input	threads/job	time to results (not counting initial queue time)	total walltime	walltime minutes
2.2.26	1x100	1	55 min	9901.5	165.0
2.2.31+	1x100	1	16 min	4123.3	68.7
2.2.31+	100x1	1	3 min	165.2	2.8
2.2.31+	100x1	8	1 min	62.3	1.1

Why so much improvement with multiple inputs?

Recommendations 4 & 5:

- **Batch your BLAST input!**
  - Current BLAST treats multiple input sequences as a single one for seeding purposes, then breaks up the results before extension
  - With many sequences, this dramatically reduces the number of database queries

- **When batching many sequences, use "–num_threads 8"**
  - Current version doesn't seem to scale above 8 threads

How is JGI doing with all that currently?

Mostly outdated versions of blast are being used

- 1104031 /global/dna/projectdirs/microbial/omics-pamm/lib/BLAST/blastall (2.2.21)
- 698394 blastall (probably 2.2.26)
- 653252 blastx
- 600898 /global/dna/projectdirs/fungal/pipeline/FRK/bin/euk/linux/blast/blastall (2.2.4)
- 274490 /global/dna/projectdirs/fungal/pipeline/2015-01-09_v1.9.1/bin/euk/linux/blast/blastall (2.2.4)
- 183805 /global/dna/projectdirs/fungal/pipeline/FRK/utils/iprscan//bin/binaries/blast/blastall (2.2.6)
- 160761 /usr/common/jgi/aligners/blast/2.2.26/bin/blastall (2.2.26)
- 138106 /global/dna/projectdirs/microbial/omics-pamm/lib/rpsblast/rpsblast (2.2.23)
- 46672 bin/blast/2.2.6/blastall (2.2.6)
- 24411 blastn
- 16832 bin/blast/2.2.19/blastall (2.2.19)
- 15190 megablast
- 13789 /projectb/sandbox/plant/iprscan/iprscan.uge/bin/binaries/blast/blastall (2.2.19)
- 12505 /usr/common/jgi/aligners/blast+/2.2.28/bin/blastn
- 10769
- 7926 /usr/common/jgi/aligners/wublast/20060510/blastp
- 5101 /global/dna/projectdirs/plant/tools/compgen/rmblast/DEFAULT/bin/rmblastn
- 2428 /usr/common/jgi/aligners/blast+/2.2.29/bin/blastn
- 2415 /usr/common/jgi/aligners/blast+/2.2.31/bin/blastn
- 1717 /global/projectb/sandbox/IMG/img/dataLoad5/tools/blast/rpsblast
- 1283 /usr/common/jgi/aligners/blast+/2.2.26/bin/tblastn

How is JGI doing with all that currently?

Many users using very large E-values

count	evalue threshold
1719	1.00E-100
2485	1.00E-50
595500	1.00E-30
41033	1.00E-20
13601	1.00E-10
4620	1.00E-08
769204	1.00E-05
121467	1.00E-04
743675	1.00E-03
165291	1.00E-02
770353	1.00E-01
692386	1
56398	1.00E+01
429	1.00E+02

count	evalue threshold
770353	1.00E-01
769204	1.00E-05
743675	1.00E-03
692386	1
595500	1.00E-30
165291	1.00E-02
121467	1.00E-04
56398	1.00E+01
41033	1.00E-20
13601	1.00E-10
4620	1.00E-08
2485	1.00E-50
1719	1.00E-100
429	1.00E+02

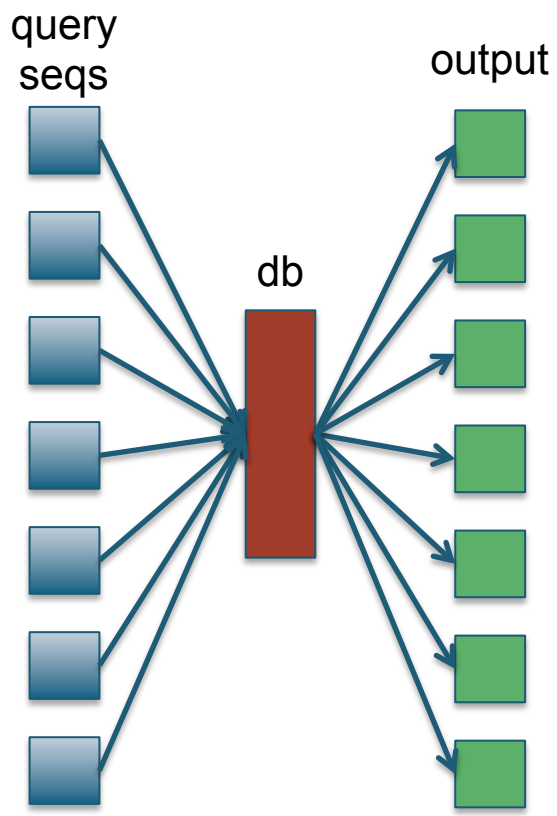
How is JGI doing with all that currently?

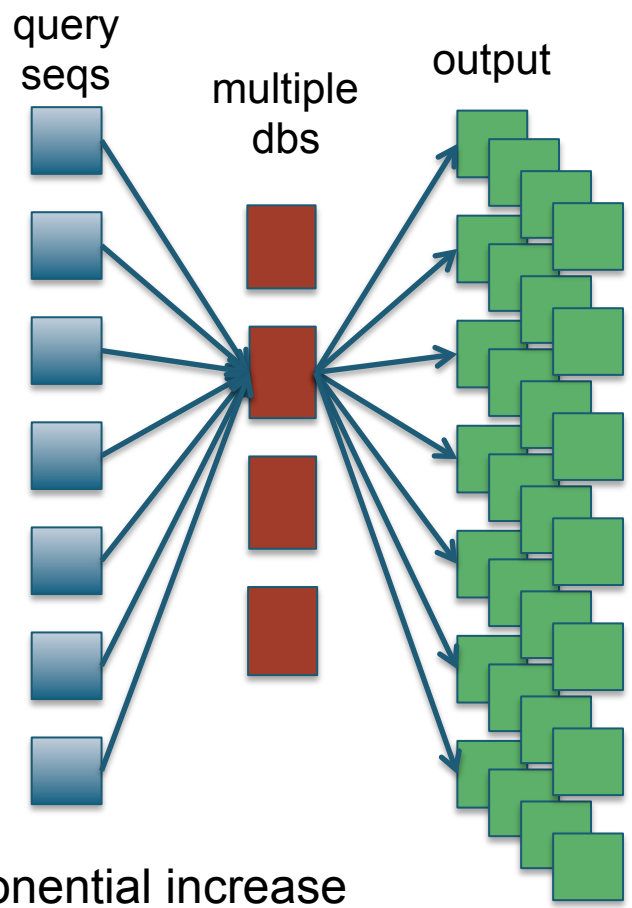
- **About half of BLAST processes are not multithreaded**

Processes	Threads
2125509	1
1327906	8
227298	4
132955	2
84781	32
70815	16
7063	5
3151	10

Thoughts on BLAST databases

- **We see some users with small, specific databases**





Exponential increase
in jobs, filesystem access,
output files…

Database reads

- **While extending and evaluating number of HSPs is the most severe bottleneck, accessing the database is a significant one**
- **Disk access time can be significant, because current version does not read all of db into memory, even if enough memory available to do so.**
- **Large job arrays may cause GPFS slowdowns**

- **Reduce disk access time by:**
  - Never run from $HOME directory
  - Stage your database directly to the compute node /scratch
  - Use one of our newly pre-staged standard dbs

Pre-staged standard BLAST databases

- **Mendel nodes have more /scratch space, and we are copying some frequently-used databases to**
  - /scratch/blastdbs
- **Working with users on plan to keep updated**

- **Current databases:**

gcontam	**unite**
mito.nt	**nt**
refseq.archaea	**refseq.bacteria**
refseq.fungi	**refseq.mitochondrion**
refseq.plant	**refseq.plasmid**
refseq.viral	

Summary of recommendations

- Ask yourself if BLAST is the best tool for the job, and consider which BLAST executable
- Use a current blast executable
- Use an appropriate E-value for your desired output
- Limit the number of output sequences
- Use multiple sequence inputs
- Use multithreading
- Combine your databases
- Use pre-staged databases, or stage your own


NERSC